data_fusion

November 17, 2021

1 Data fusion experiment

1.1 Setup

[1]:
```
# Scikit-Learn  0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

import pandas as pd
import open3d as o3d
import numpy as np
import math

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
%matplotlib inline
# %matplotlib auto
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)


from pandas import DataFrame
# For plotting
import plotly.io as pio
import plotly.graph_objects as go
```

```
# Where to save the figuresimport pandas
from pandas import DataFrame


# For plotting
import plotly.io as pio
import plotly.graph_objects as go

import numpy as np
PROJECT_ROOT_DIR = "."
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images")
DATA_PATH = os.path.join(PROJECT_ROOT_DIR, "data")
Trial_data_path = os.path.join(DATA_PATH, "experiment_trail_data")
os.makedirs(IMAGES_PATH, exist_ok=True)
```

1.2 Load raw data

[2]:
```
meltpool_geometry_file = os.path.join(Trial_data_path, "geometry.csv")
position_file = os.path.join(Trial_data_path, "position.csv")
velocity_file = os.path.join(Trial_data_path, "velocity.csv")


# pointsSegDF = pandas.read_csv(PCLFile_seg, sep=' ', names=('x', 'y', 'z'),␣
 ↪skiprows=[i for i in range(0,11)])
data_meltpool = pd.read_csv(meltpool_geometry_file, sep=",", header=None,␣
 ↪skiprows=1)
data_position = pd.read_csv(position_file, sep=",", header=None, skiprows=1)
data_velocity = pd.read_csv(velocity_file, sep=",", header=None, skiprows=1)


data_meltpool.columns = ["time", "stamps", "major_axis", "minor_axis",␣
 ↪"orientation", "x", "y", "minor_axis_average"]
data_meltpool.head()
```

[2]:

	time	stamps	major_axis	minor_axis	orientation	\
0	1.620046e+18	1.620046e+18	75.940765	9.707503	1.278871	
1	1.620046e+18	1.620046e+18	74.889854	9.638904	1.285647	
2	1.620046e+18	1.620046e+18	65.562218	9.731025	1.285165	
3	1.620046e+18	1.620046e+18	79.430443	9.566442	1.284775	
4	1.620046e+18	1.620046e+18	50.400818	9.927440	1.241938	

	x	y	minor_axis_average
0	313.863678	187.663574	2.517748
1	314.410522	189.504410	2.517748
2	315.190430	191.549301	2.517748

```
3  313.929382  187.706131            2.517748
4  316.259277  196.590469            2.517748
```

[3]:
```
data_velocity.columns = ["time", "stamp", "speed", "vx", "vy", "vz"]
data_velocity.head()
```

[3]:

	time	stamp	speed	vx	vy	vz
0	1.620046e+18	1.620046e+18	0.0	0.00000	0.0	0.0
1	1.620046e+18	1.620046e+18	0.0	0.00001	0.0	0.0
2	1.620046e+18	1.620046e+18	0.0	0.00000	0.0	0.0
3	1.620046e+18	1.620046e+18	0.0	0.00000	0.0	0.0
4	1.620046e+18	1.620046e+18	0.0	0.00000	0.0	0.0

[4]:
```
data_position.columns = ["time","stamp", "x", "y","z"]
data_position.head()
```

[4]:

	time	stamp	x	y	z
0	1.620046e+18	1.620046e+18	-1.320345	0.247014	0.664305
1	1.620046e+18	1.620046e+18	-1.320344	0.247014	0.664305
2	1.620046e+18	1.620046e+18	-1.320344	0.247014	0.664305
3	1.620046e+18	1.620046e+18	-1.320345	0.247014	0.664305
4	1.620046e+18	1.620046e+18	-1.320344	0.247014	0.664305

1.3 Data visualization

[5]:
```
# Visualizing 5-D mix data using bubble charts
# leveraging the concepts of hue, size and depth
'''
fig = plt.figure(figsize=(14, 10))
ax = fig.add_subplot(111, projection='3d')
t = fig.suptitle('Multi-dimensional data visualization', fontsize=14)

xs = list(data_position['x'])
ys = list(data_position['y'])
zs = list(data_position['z'])
data_points = [(x, y, z) for x, y, z in zip(xs, ys, zs)]

ss = list(data_meltpool['minor_axis'])
colors = [list(data_velocity['speed'])]

for data, color, size in zip(data_points, colors, ss):
    x, y, z = data
    ax.scatter(x, y, z, alpha=0.4, c=color, edgecolors='none', s=size)
```

```
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
'''
```

[5]:
```
"\nfig = plt.figure(figsize=(14, 10))\nax = fig.add_subplot(111,
projection='3d')\nt = fig.suptitle('Multi-dimensional data visualization',
fontsize=14)\n\nxs = list(data_position['x'])\nys = list(data_position['y'])\nzs
= list(data_position['z'])\ndata_points = [(x, y, z) for x, y, z in zip(xs, ys,
zs)]\n\nss = list(data_meltpool['minor_axis'])\ncolors =
[list(data_velocity['speed'])]\n\nfor data, color, size in zip(data_points,
colors, ss):\n    x, y, z = data\n    ax.scatter(x, y, z, alpha=0.4, c=color,
edgecolors='none', s=size)\n\n    \n
\nax.set_xlabel('x')\nax.set_ylabel('y')\nax.set_zlabel('z')\n"
```

[23]:
```
fig = go.Figure()

# Plot points, using distance FROM plane as color scale
fig.add_trace(go.Scatter3d(x = data_position['x'],
                           y = data_position['y'],
                           z = data_position['z'],
                           mode='markers',
                           marker=dict(size=data_meltpool['minor_axis']/5,
                                       color=data_velocity['speed']␣
↪[data_velocity['speed'] > 0.0001], #data_meltpool['minor_axis']/5 or␣
↪data_velocity['speed']
                                       colorscale='oranges',# other colorscale␣
↪options: agsunset
                                       cmin = 0,  # minimum color value, ---␣
↪data_velocity['speed'].min()   ### need some adjustment here
                                       cmax = 0.006,  # maximum color value
                                       showscale=True)
                          )
            )


fig.update_layout(scene = dict(xaxis = dict(nticks=6, range=[data_position['x'].
↪min(), data_position['x'].max()]),
                               yaxis = dict(nticks=6, range=[data_position['y'].
↪min(), data_position['y'].max()]),
                               zaxis = dict(nticks=3, range=[data_position['z'].
↪min(), data_position['z'].max()]),
                               aspectmode = 'data'  # preserve the proportion␣
↪of actual axes data
                               )
                  )
```

```
# Use the offline mode of plotly:
pio.write_html(fig, file="data_visualization.html", auto_open=True)
```

[ ]: